Affect and Learning: a computational analysis

Broekens, D.J.

4

Affect and Thought

Affect-Controlled Simulation Selection

In this chapter we study affective control of the amount of simulated anticipatory behavior in artificial adaptive agents. Artificial affect is positive when an agent is doing better than expected and negative when doing worse than expected, as defined in Chapter 2 and used in the study in Chapter 3. Our approach is based on model-based Reinforcement Learning (although we use a different model than the one used in Chapter 3) and inspired by the *Simulation Hypothesis* (Cotterill, 2001; Hesslow, 2002). In contrast to the research described in Chapter 3, where we used affect to control the exploration – exploitation rate directly, in an adaptive agent that has a purely reactive architecture (no internal simulation of interaction), here we study *the adaptiveness of an artificial agent, when action-selection bias is induced by an affect-controlled amount of simulated anticipatory behavior*. To this end, we introduce an affect-controlled *simulation-selection* mechanism that selects anticipatory behaviors for simulation from the agent's Reinforcement Learning model.

Based on experiments with adaptive agents in two nondeterministic partially observable grid worlds we conclude that (1) internal simulation has an adaptive benefit and (2) affective control reduces the amount of simulation needed for this benefit. This is specifically the case if the following relation holds: positive affect decreases the amount of simulation towards simulating the best potential next action, while negative affect increases the amount of simulation towards simulating all potential next actions. Thus, agents "feeling positive" can think ahead in a narrow sense and free-up working memory resources, while agents "feeling negative" must think ahead in a broad sense and maximize usage of working memory. Our results are consistent with several psychological findings on the relation between affect and learning, and contribute to answering the question of *when* positive versus negative affect is useful during adaptation.

4.1 Introduction

In this Chapter we study affective control of the amount of information processing in artificial adaptive agents. In order to model affective control of information processing, we use the measure for artificial affect, as defined in Chapter 2, which relates to an adaptive agent's relative performance on a learning task. Artificial affect measures how well the agent improves. Our adaptive agent learns by reward and punishment. Thus we define "wellness" based on averages over reinforcement signals. As such, the agent's performance is defined by the difference between the long-term average reinforcement signal ("what am I used to") and the short-term average reinforcement signal ("how am I doing now") (cf. Schweighofer & Doya, 2003). Our measure of artificial affect thus relates to natural affect in the sense that it characterizes the situation of the agent on a scale from good to bad. Further, as our measure is based on average reinforcement signals, it relates more to mood than emotion.

We have developed a variation to the model-based Reinforcement Learning (RL) paradigm (Sutton & Barto, 1998). This variation enables us to view information processing in light of the *Simulation Hypothesis* (Cotterill, 2001; Hesslow, 2002). The Simulation Hypothesis states that thinking is internal simulation of behavior using the same sensory-motor systems as those used for overt behavior (Hesslow, 2002). The main reason for adopting the Simulation Hypothesis is that it argues for evolutionary continuity between agents that consciously think and agents that do not. We believe that evolutionary continuity is a critical aspect in studying behavior, emotions, consciousness and cognition. In this chapter, we refer to simulation as described by the Simulation Hypothesis.

An important current issue is how simulation of interaction is integrated with real interaction while using the same mechanisms (see models by, e.g., Shanahan, 2006; van Dartel & Postma, 2005; Ziemke, Jirenhed & Hesslow, 2005). Our agents are able to internally simulate anticipatory behavior using their RL model. The agent thinks ahead by selecting one or more potential next action-state pairs for internal simulation. This action-state and its associated value are fed into the RL model as if these were actually observed. This introduces a bias to predicted values. Our action-selection mechanism uses these biased values to select the agent's next action. Subsequently, the values are reset to the original values before simulation. Thus, internal simulation temporarily biases the predicted values in the RL model, thereby biasing action selection.

In this chapter we report on a study on *the adaptiveness of an artificial agent, when action-selection bias is induced by an affect-controlled amount of simulated anticipatory behavior.* Thus, the main contributions of this chapter to the affect-learning and Simulation Hypothesis literature are:

- The introduction of an affect-controlled mechanism for the selection of internally simulated behavior instead of actual behavior; we define this mechanism as *simulation selection*.
- A study into the influence of affect on learning, when used to control the amount of internally simulated interactions, where simulated interactions bias actual action selection. As we use internal simulation as a model for

information processing, we investigate affect as a modulator for the trade-off between internal versus external information processing effort (Aylett, 2006).

In Section 4.2 we review the relation between internal simulation and our approach in more detail. In Section 4.3 we present our computational model and how it implements artificial affect, internal simulation of behavior and learning. In Section 4.4 we describe our experimental setup. In Section 4.5 we present experimental results. In Section 4.6 we discuss our approach in a broader context.

4.2 Internal Simulation of Behavior as a Model for Thought

Our approach towards anticipatory simulation is inspired by the Simulation Hypothesis stating that conscious thought consists of “simulated interaction with the environment” (Hesslow, 2002). Thoughts consist of internally simulated chains of interaction with the environment and evaluation of those simulated interactions. As such, thoughts are virtual versions of real interactions. For this to be possible, a brain must be able to internally simulate actions, perceptions and evaluations of action-perceptions in an off-line manner. That is, the brain has to simulate potential interaction with the environment while simultaneously controlling the body such that it is able to successfully interact with the environment. Hesslow (2002) and Cotterill (2001) provide extensive evidence for the biological and psychological plausibility of such a simulation process.

4.2.1 Thought and Internal Simulation of Interaction

In addition to being plausible, internal simulation of behavior is also a convenient model for thought, especially in the context of adaptive behavior and evolutionary continuity. First, if an agent is able to internally simulate a certain interaction, this simulation can reactivate the value of that interaction and thereby (1) influence decision making with predictions based on previous experiences and (2) enhance learning by propagating the value of that interaction to other related interactions. Second, the Simulation Hypothesis is said to provide a bridge between species that consciously think and those that do not (Hesslow, 2002): no additional mechanisms are needed for thought, apart from those that enable off-line simulation of interaction.

Recently, strong evidence for a link between internal simulation, adaptive behavior and evolutionary continuity has been presented. Foster and Wilson (2006) showed that awake mice replay in reverse order behavioral sequences that led to a food location; a crucial finding for the above mentioned link. First, it suggests that mice are able to internally simulate interaction with the environment, showing that simulation mechanisms need not be restricted to humans. This supports the possibility of evolutionary continuity of the human thought process. Second, internally replaying a sequence of interactions can potentially increase learning in mice in the same way as *eligibility traces* can enhance learning in Reinforcement Learning (Foster & Wilson, 2006). An eligibility trace (see Sutton & Barto, 1996) can be seen as a sequence of recent interactions with the environment. Delayed reinforcement is distributed over all the interactions stored in the trace. This mechanism can dramatically increase learning performance of simulated adaptive agents, and therefore provides a plausible argument for an immediate benefit of internal simulation (different from benefits related to complex cognitive abilities such as planning).

4.2.2 Working Memory, Simulation Selection and Internal Simulation of Behavior

If a thought is an internally simulated interaction, and working memory (WM) contains the thoughts of which we are consciously aware, then WM contains a set of currently maintained internally simulated interactions—specifically the episodic buffer that is a multi-modal limited-capacity storage buffer (Baddeley, 2000). Further, for a specific thought to enter WM, it is often assumed that the thought has to be active above a certain threshold (exemplified by a computational neuronal model by Dehaene, Sergent and Changeux (2003)).

The "internal simulation thought process" would go like this. An agent in a specific situation starts to pay attention to several situational aspects. These aspects start entering the central executive of working memory (Baddeley, 2000) and are thereby above threshold. Now, the central executive pushes a multi-modal simulation of future (or related) interactions from long term memory to the episodic buffer, where it is maintained. As the episodic buffer has limited capacity, the interaction can reside in the buffer until being replaced (pushed away) by new simulated interactions. Thus, filling the buffer depends (among other things) on how critical the filter (central executive) is in passing information to the buffer. The episodic buffer is filled with those internally simulated interactions that are attended to with sufficient intensity. Therefore, the higher the simulation-selection threshold, the smaller the amount of internally simulated behaviors maintained in the episodic buffer.

Interestingly, if thought is internal simulation of behavior using the same sensory-motor mechanisms as real behavior, then the selection of those thoughts should resemble the selection of behaviors. Action-selection has been defined as the problem of continuously deciding what action to select next in order to optimize survival (Tyrell, 1993). "Thought selection", to which we refer as *simulation selection*, can therefore be defined in a similar way. Simulation selection is the problem of continuously selecting behaviors for internal simulation such that action selection is assisted, not hindered. The latter is critical as, according to the Simulation Hypothesis, action selection and simulation selection should be tightly coupled: both use the same mechanisms. Errors in simulation selection can directly influence action-selection and thereby be responsible for actions that are erroneous too. In our computational model we introduce a simulation-selection component based on precisely these principles. Moreover, the simulation-selection threshold in our model is dynamically controlled by artificial affect (Section 4.3.2, 4.3.3).

4.3 Model

In this section we explain the computational model used to study the main question. We use adaptive agent based modeling. Our agents "live" in grid worlds. Figure 4.1 shows the overall architecture of our computational approach.

The affect mechanism calculates artificial affect based on how well the agent is doing compared to what it is used to. The simulation-selection mechanism selects next interactions for simulation, using a threshold controlled by artificial affect. The threshold filters which potential next interactions are simulated and which not. Selected interactions are fed into the RL model (as if they were real). This biases predicted values of states in the RL model. The action-selection mechanism selects an action based on these biased values using a greedy algorithm. The action is executed, and the agent perceives the next state. Our approach is related to *Dyna* (Sutton, 1990). In the general discussion we explore some of the similarities and differences.

We first discuss the components of the model and the way it learns using RL principles. Then we explain how we have implemented the Simulation Hypothesis on top of our model. Subsequently we explain how artificial affect is used to control the amount of internal simulation the agent uses to bias the predicted values employed by its action-selection mechanism. Finally, we explain how the action-selection mechanism integrates everything.

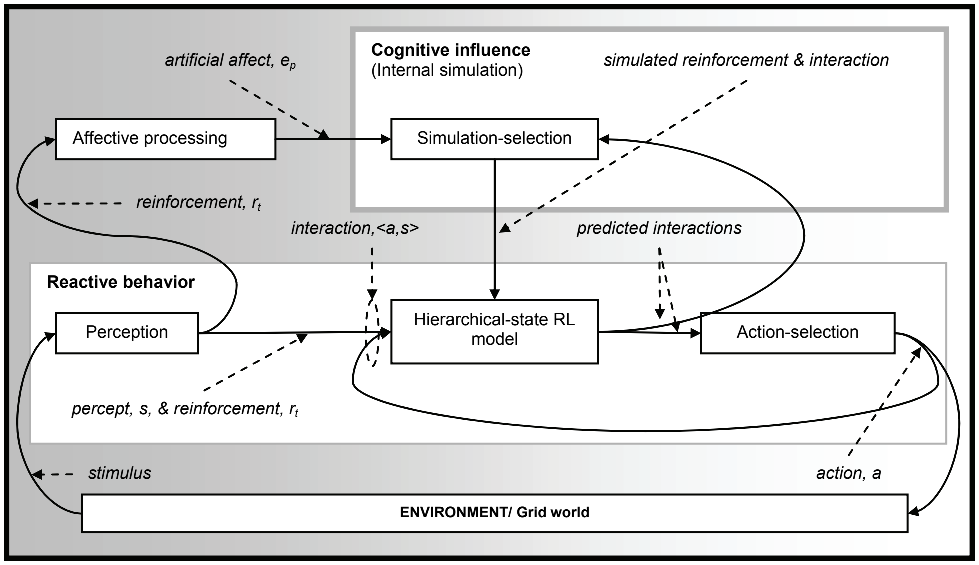

Figure 4.1. Overview of the different components in our model. Components are detailed below.

4.3.1 Hierarchical State Reinforcement Learning (HS-RL): A Variation of Model-Based RL

Our model is a combined forward (predictor) and inverse (controller) model for learning agent behavior (Demiris & Johnson, 2003). The model learns to predict the next state given the current state and an action, enabling forward simulation of interaction. At the same time it learns to predict the values for potential next actions, enabling agent control. Basically, the agent's memory structure is a directed graph that is learned by interaction with the environment. Two types of nodes exist: (1) nodes that encode <*a*, *s*> tuples, where *s* is an observed state and *a* the action leading to that state, and (2) nodes that encode (h_l, <*a*’, *s*’>) tuples. Here, h_l is a history of observed action-state pair transitions <a^{t-l}, s^{t-l}><a^{t-l+1}, s^{t-l+1}>…<a^{t-1}, s^{t-1}> with *l* the history length not greater than a maximum length *k*, and <*a’*, *s’*> = <a^t, s^t> the action-state pair predicted by history h_l at time *t*. The existence of type 1 nodes depends on the states experienced by the agent. The existence of type 2 nodes, and the connectivity between type 1 and type 2 nodes depend on observed transitions from <*a*, *s*> to <*a’*, *s’*>. Thus, the memory is initially empty and is constructed while the agent interacts with its environment; our agent learns online. We thus assume *certainty equivalence*. This is closer to real life than a forced separation between exploration and exploitation phases, even though the model might be highly suboptimal at the start (Kaelbling, Littman & Moore, 1996).

The model is constructed as follows. The agent selects an action, $a \in A$, from its set of potential actions, A, using the action-selection mechanism (Section 4.4). It executes the action and perceives the result, s. A type 1 node <a, s> is created *if and only if there does not exist such a node* <a, s>. Consider, for example, an agent that has chosen some action $\tilde{a}$ and experiences some state σ. Because its model does not yet contain a node that represents <$\tilde{a}$, σ> it is created (e.g., s_1 in Figure 4.2a). Note that we use s_i (indexed) to refer to <a, s> tuples (type 1 nodes) instead of s to refer to observed states. Now the agent selects and executes a new action, resulting in a new situation s_2=<$\tilde{a}$', σ'>, giving a new node that represents s_2 (Figure 4.2b). To model that s_2 follows s_1 (s_1 predicts s_2), the previous situation, s_1, is now connected to the current situation, s_2, by creating a new type 2 node, defined as an *interactron* (sic!), connected to s_1 and s_2 with edges as shown in Figure 4.2c. This node I_1 thus encodes (h_1, s_2) with h_1 being the history of length 1 before the transition to action-state pair s_2,, in our example h_1=s_1. This process continues while exploring and the process is applied hierarchically to all active nodes. A type 1 node is active if the current situation <a^t, s^t> equals the <a, s> tuple encoded by that node. A type 2 node (h_l, <a', s'>) is active if and only if h_l equals the most recent observed history <a^{t-l}, s^{t-l}><a^{t-l+1}, s^{t-l+1}>…<a^{t-1}, s^{t-1}> and the prediction <a', s'> equals <a^t, s^t>. For example, node I_1 and s_2 in Figure 4.2c are active. An additional example is presented in Figure 4.2d and 4.2e. If situation s_2 is followed by a new situation s_3, the resulting memory structure is shown in Figure 4.2d, with active nodes s_3, I_2 and I_3. If, on the other hand s_2 is followed by s_1, the resulting structure is shown in Figure 4.2e, with active nodes s_1, I_2 and I_3. Note that the maximum length of a history encoded by a node is bounded by k, therefore the maximum number of active type 2 nodes is k (for computational reasons k = 10 in this study; for more on k see below and Broekens & DeGroot, 2004b).

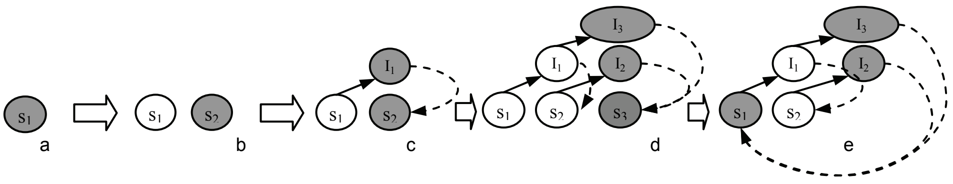

Figure 4.2a-e. Examples of the agent's memory structure

Every node (h_l, <a', s'>) has three properties r, v, and υ, with r the reward and v the value (a.k.a. Q-value) of the tuple (h_l, <a', s'>), and finally υ is a statistic for the transition probability between h_l and <a', s'>. If at a later time the sequence of situations $h_l s_i$ is *again* observed by the agent, then the statistic υ of the type 2 node encoding the tuple (h_l, s_i) is incremented— υ is a counter that is initially zero and represents the *usage* of an interactron. Thus, υ can be used to calculate the transition probability $p(s_i \mid h_l)$ using the following more generic formula:

$$p(x \mid y) = \upsilon_x \Big/ \sum_{i=1}^{|X_y|} \upsilon_{x_i} \ , \qquad (4.1)$$

where y is a node encoding (h_{l-1}, <a, s>) with h_l=$h_{l-1}s_y$ and s_y=<a, s>, and $x \in X_y$. Here $X_y = \{x_1,\ldots,x_n\}$ is the set of interactron nodes that encode (h, <a', s'>) tuples and are predicted by y, x is the node (h_l, s_i) of which we want to know the transition probability $p(s_i \mid h_l)$, and υ_x and υ_{x_i} are the counters belonging to x and x_i respectively. This function calculates the conditional probability of observing an action-state pair <a, s> after a history of action-state pairs h_l using the most recent model of the world.

Furthermore, we define a global threshold called the *forgetting rate*, θ, representing the minimal "survival probability" for an interactron. If $p(x \mid y)<\theta$, the corresponding interactron x is forgotten and removed from the memory, including all of its predictions. In this manner the stability of an agent's long-term memory is modeled, and it corresponds to Bickhard's (2000) notion of interaction (de)stability based on consistent confirmation of predicted interactions. The relation between interaction (de)stability and our learning model is explained in more detail in (Broekens & DeGroot, 2004b). In our experiments we use θ to vary the speed with which the agent forgets knowledge.

To learn based on reinforcement, every interactron has a value v, with:

$$v = r + \gamma v_{next} \text{, with } v \text{ maxed-out such that } \min(r, v_{next}) \le v \le \max(r, v_{next}) \quad (4.2)$$

where r is the learned reward for a certain interactron, γ the discount factor (equal to 1.0 in all our experiments, see below for why this does not pose a problem in our approach) and v_{next} is a back-propagated value from next predicted future states. As multiple nodes can be active at the same time, these nodes learn simultaneously. Several steps are involved. First, all k active interactrons are reinforced by a signal from the environment, r_t, at time t. For every such interactron y, its learned reward $r(y)$ is adapted according to the formula:

$$r(y)^{t+1} = r(y)^t + \alpha(r_t - r(y)^t) \ , \qquad (4.3a)$$

where α is the agent's learning rate. Second, for every interactron y, $v_{next}(y)$ is calculated as follows:

$$v_{next}(y)^{t+1} = \sum_{i=1}^{|X_y|} v(x_i \mid y)^t \times p(x_i \mid y)^t \ , \quad (4.3b)$$

where $v(x_i \mid y)^t$ is defined as the value of interactron x_i, with x_i predicted by y. This indirect part of an interactron's value is thus the weighted average of the values belonging to the interactrons X_y that represent the situations that y predicts, where the weighting is according to the probabilities $p(x_i \mid y)^t$ at time t over all i. Note that only active nodes y are updated, i.e., we use lazy propagation.

In an agent control setting, the model can be summarized as follows. At every step, all active interactrons predict potential next situations, at most k of these interactrons can be active, and the 1st to kth interactron predicts potential next action-state pairs <*a*', *s*'> using a history of length 1 until k respectively (e.g., I_3 is a k=2 interactron with history s_1s_2). As such, this memory learns 1st...kth order Markov Decision Processes (MDPs) in parallel. This property enables it to cope with partially observable worlds in which the partial observability can be resolved using at most a history of length k. At most k MDPs are active at the same time, with some of them predicting the future based on little history and some predicting the future based on a history of length at most k. The predictions consist of estimated future values for next action-state pairs, as usual. However, k of these MDPs are active at the same time, so action selection integrates not over the predictions of 1 such MDP but over the predictions of k such MDPs. How action selection integrates over these parallel predictions is explained in the section on action selection below. Note that our model underuses the Markov property, as it keeps track of, and constructs nodes for, all history up to k steps back *all the time, not only when a certain history is actually needed to solve the partial observability of the world*. For an interesting approach that relates to ours and that proposes some solutions for better using the Markov property see McCallum's (1995) *utile suffix memory*.

An important difference between our approach and many other model-based RL approaches is that our MDPs have a maximal length of k steps and nodes only propagate values to their own history. On the one hand this is a benefit in that reward/value propagation is never cyclic. Values are propagated back through multiple, partially overlapping k-finite MDPs. This makes our model particularly robust in cyclic learning tasks (even for cycles smaller than k steps): our world model forces values to propagate from a well-defined end with a long history to a well-defined beginning with no history, the values are *not* recursive. As a result, in our model the discount factor can be equal to 1.0. On the other hand this characteristic also poses a problem, as values further than k steps away cannot be propagated back, resulting in the need for regular reward intervals. This could be resolved (at the expense of cyclic-task robustness) by allowing values to propagate *not only* to nodes encoding for a shorter history at the previous timestep but *also* to nodes encoding for a history of equal length at the previous timestep, effectively making values recursively defined. That is, a node $s_1h_{l-1}s_t$ encoding for a situation s_t with a history s_1h_{l-1} of length l not only propagates its value to a node $s_1h_{l-2}s_{t-1}$ with $h_{l-1}=h_{l-2}s_{t-1}$, but also to a node $s_0s_1h_{l-2}s_{t-1}$. Other limitations, experimental convergence results as well as several choices for the world model itself are discussed in more detail in Broekens & DeGroot (2004b).

To summarize; with every step of the agent, our model updates (1) the world model, (2) its statistics and rewards, and (3) the values. A maximum of *k* nodes is updated at every step. Every node encodes the current action state, an action-state history equal to the most recent action-state history, a reward, a value and a usage statistic. In the ideal (policy unbiased) case, the value of every such node converges as is usual for *Q*-values in RL.

4.3.2 Internal Simulation of Behavior: a Temporary Bias to Predicted Action-State Values

We now explain how internal simulation of action-state pairs (a.k.a. interactions/situations) temporarily biases the predicted value of next actions, and thereby influences action selection. Instead of action selection, the following steps are involved:

1. S*imulation selection*: at time *t* select a subset of to-be-simulated interactions (action-state pairs) from the set of interactions predicted by all *k* active interactrons.
2. *Simulate*: use a selected interaction from that subset as if it was a real interaction. The agent's memory advances to time *t*+1. As this is a simulation step, we lack the reinforcement signal r_t that accompanies real interactions. Instead, r_t is simulated using the value, *v*, of the simulated interaction. We simulate a predicted interaction and its associated value as if they were both real.
3. *Reset state*: to be able to select an appropriate action in Step 4, reset the memory's state (the active nodes) to the previous timestep, i.e., time *t*. The net effect of Step 2 and 3 is that, due to the value propagation mechanism, a temporary bias—based on future predictions at *t*+1—is introduced to the value of predicted next interactions. Step 2 and 3 are repeated for every to-be-simulated interaction. These biased values are reset in Step 5 (after action-selection in Step 4). If we would keep this bias after action selection, it would

break our model (in RL the reward *r* must be used to make the value *v* converge; using v_{t+1} to converge *v* introduces a problem of cumulative prediction errors).

4. *Action selection*: select the next action using the mechanism explained in Section 4.3.4. Thus, the propagated values of the simulated predicted interactions directly bias action selection. Our anticipation mechanism is best understood as *state anticipation* (Butz, Sigaud & Gerard, 2003).
5. *Reset values*: reset the reinforcement related variables *v, r* and v_{next} of the interactions that were changed at Step 2 (simulation) to the values of *v, r* and v_{next} of these interactions before Step 2.

In the studies reported in this chapter, simulation is bounded to a depth of 1, i.e., anticipation is just one step ahead. However, our simulation mechanism can easily support the simulation of multiple time steps ahead by processing Step 1 to 3 backwards from *t*+*d* to *t*+1 in all possible branches of potential next interactions, with *d* the simulation depth. Now, action selection at time *t* is biased by accumulated simulated values of interactions up to *d* steps ahead. A potential problem is the build-up of small prediction errors. This invalidates the values of next actions, and action selection could be severely compromised. To enable multi-step simulation, accumulation of prediction errors during multi-step simulation should be investigated (e.g., Hoffmann & Möller, 2004).

Step 1 is the s*imulation-selection* mechanism and selects predicted interactions to be simulated. This is a critical component in our simulation mechanisms as it defines the amount of internally simulated information per time step. In our experiments we use four static simulation-selection mechanisms and several dynamic ones (also referred to as *simulation strategies*):

- Static simulation selection: sort anticipated interactions according to their predicted value. Select a number of the best anticipated states for simulation. The selected interactions are sent to the model for simulation (Step 2).
- Dynamic simulation selection: again, anticipated interactions are sorted according to their predicted value. In contrast to static selection, here affect is used to control the amount of predicted interactions that are selected from the sorted list. We explain this in Section 4.3.3.

In essence, simulation selection is controlled by a simulation-selection threshold*,* t_s*,* of a t_s-Winner-Take-All (WTA) simulation selection ranging from infinite (no simulation) to zero (select and simulate all predicted action-state pairs). This threshold is used by the simulation-selection mechanism to filter the set of predicted interactions that are simulated, i.e., to select potential next behaviors for processing in working memory. Our simulation-selection mechanism uses t_s in the following way: t_s defines the percentage of *predicted best next interactions* that should be internally simulated (so in a sense it is an inverse threshold). If $t_s < 0$ (overly selective threshold), no simulation is done. If $t_s \geq 0$ (selective threshold) only the interaction with the highest predicted value is simulated, if $t_s \approx 1.0$ (non-selective threshold) all interactions are simulated. The final result of simulation can be summarized as follows: anticipatory simulation introduces a bias to the values of the set of predicted next possible action-state pairs, thereby influencing the result of action selection. In the next section we explain how artificial affect is used to dynamically set the threshold t_s, instead of statically (Broekens, 2005).

4.3.3 Affective Modulation of WM Content: Affect Controls the Amount of Internal Simulation

Using the measure for artificial affect, e_p, introduced in Chapter 2, it has now become straightforward to model affective control of the amount of internal simulation (i.e., affective control of working memory content), the basis of our study. Control can be modeled in several, equally plausible, ways. By equating the simulation-selection threshold, t_s, to $1- e_p$, it varies between 0 and 1 depending on affect being positive or negative respectively. This reflects the hypothesis that positive affect decreases the amount of internal simulation favoring narrow, exploitative thoughts (i.e., only action-state pairs with a high value are internally simulated), while negative affect increases the amount of simulation favoring broad thoughts, including explorative ones (i.e., action-state pairs with low values are also simulated). This relates to results found by Rose et al. (1999). In our model this means that happy agents (i.e., performing better than expected) simulate positive thoughts, while a discontent agent simulates many thoughts including negative ones. So:

$$t_s = 1 - e_p \qquad (4.5)$$

Second, we hypothesize the inverse relation, that is, negative affect decreases the amount of simulation while positive affect increases the amount of action-state pairs that can enter working memory for simulation:

$$t_s = e_p \qquad (4.6)$$

Now, positive affect *increases* the thought-action repertoire (Ashby et al., 1999). This relates to results found by Goschke and Dreisbach (2004).

A third hypothesis is that the intensity of affect controls the amount of simulation, instead of the positiveness and negativeness of affect. Here, intense is either negative affect ($e_p \approx 0$) or positive affect ($e_p \approx 1$) while not intense is neutral ($e_p \approx 0.5$). If affect is intense, simulate a lot (reflecting the fact that significant changes occurred that might need extra processing (Scherer, 2001)). If affect is not intense, do not simulate a lot. Note that intensely positive or negative does not necessarily mean arousing, arousal is considered out of scope for this thesis. The simulation-selection threshold is:

$$t_s = 2 \times abs(0.5 - e_p) \qquad (4.7)$$

And, as a control condition, the inverse relation is:

$$t_s = 1 - 2 \times abs(0.5 - e_p) \qquad (4.8)$$

In Section 4.5 we report on the results of a systematic study that investigated the influence of internal simulation on the adaptiveness of artifical agents, when the amount of simulation is modulated by affect. Modulation is according to the hypotheses mentioned above.

4.3.4 Integrating Everything: Greedy Action Selection over Biased Value Predictions

In our approach, action selection must integrate over the predictions of at most k MDPs in parallel: action selection integrates over the action-state values as predicted by all k active nodes, each node representing a possible "current state". This is an important difference with standard model-based RL as such models typically use the values for next actions as predicted by one "current state" (see, e.g., Kaelbling, Littman & Moore, 1996). As a result, our action-selection mechanism is slightly different. It is inspired by parallel inhibition and excitation of actions in the agent's set of actions, A. The inhibition/excitation originates from the k active interactrons and is calculated as follows:

$$l(a)^t = \sum_{i=1}^{k} \sum_{j=1}^{|X_{y_i}|} v(x^i{}_j \mid y_i)^t \times p(x^i{}_j \mid y_i)^t \ , \qquad (4.9)$$

where $l(a)^t$ is defined as the level of activation of an action $a \in A$ at time t, and y_i an active interactron at time t. Further, x^i_j must predict action a. Therefore, x^i_j=(h, <a, s>) with $h = h(y_i)s_{y_i}$ and $(h(y_i), s_{y_i}) = y_i$ and $s_{y_i} =< a^t, s^t >$. This clause enforces that any of the action-state pairs that are predicted by any of the *k* active interactrons should inhibit (negative value) or excite (positive value) the corresponding action, *but not other actions*.

Finally the action *a* to be executed is such that:

$$l(a)^t = \max\left(l(a_1)^t, ..., l(a_{|A|})^t\right) \qquad (10)$$

If there are only bad actions (i.e., $l(a)^t$<0) a weighted stochastic selection based on $l(a_1)^t$,…,$l(a_{|A|})^t$ is made instead; the action with the highest activation has proportionally the highest chance of being chosen resulting in a probabilistic Winner-Take-All action-selection. As such, action selection uses a super-threshold greedy selection with sub-threshold linear weighted stochastic selection.

Further, depending on when the action-selection mechanism is invoked it either uses unbiased (before simulation) values to select the next action, or biased (after simulation) values to select actions. This allows us to address the main question of our study: what happens if action-selection bias is induced by an amount of simulated anticipatory behavior, and if this amount is dynamically controlled by artificial affect?

To wrap up this section on the computational model consider the following. The number of thoughts that occupy working memory is often interpreted as an indicator of the intensity of information processing. As a thought equals an internally simulated behavior in our model, and the number of thoughts that occupy working memory equals the amount of internally simulated behavior, it is now clear that we indeed study affective-control of information processing.

4.4 Method

To investigate the influence of affect-controlled anticipatory simulation of future action-state pairs, we have set up a grid-world environment consisting of walls, roadblocks, cues, food and empty spaces. We use two non-deterministic (i.e., changing), partially-observable grid worlds. Common to our two grid worlds is that the agent *can* walk on walls, but is discouraged to do so, which is why we call our "wall" "lava" (reinforcement *r*=–1.0). The agent moves around by selecting an action *a* from the set of possible actions *A*={*up*, *down*, *left*, *right*}, and observing its immediate surroundings (not its position) using a four-neighbor-plus-center metric just after executing the action. This is an <*a*, *s*> tuple as defined in the model (Section 4.3).

The first grid world is taken from (Broekens & Verbeek, 2005), and aims to test how well agents using different simulation strategies can cope with a sudden change in both reward and world structure (Figure 4.3). In this world, the agent (black square) learns to cope with two alternating goal and start locations (‘f’=food, reinforcement *r*=1.0). Alternation is random and after every trial. A trial ends when the agent has found the goal: the agent is put back at a randomly chosen start location after having reached the randomly chosen goal location. The total number of trials to learn a task is 500. We define such sequence of 500 trials as a *run*. Additionally, at trial 250, the world is changed in the following way. Two negatively reinforced roadblocks (‘b’=block, *r*=–0.5) are placed in front of the goal locations, and the food reward is increased to 1.75 to compensate for the roadblocks. As a result, both the world and the reward structure of that world change. The agent is, of course, unaware of this change, and, as our model learns lazily, no value updates or world-model changes are made. The agent has to learn these new characteristics of the world. We call this grid world the *switch-to-invest* grid world, as it is constructed to measure how an agent copes with a change in the environment that introduces an investment to be made before an otherwise easily obtainable goal.

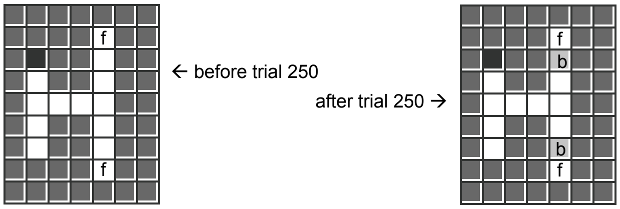

Figure 4.3. Switch-to-invest task. Potential start locations are alternated between the top-left and bottom-left arms, goal locations are alternated between the top-right and bottom-right arms.

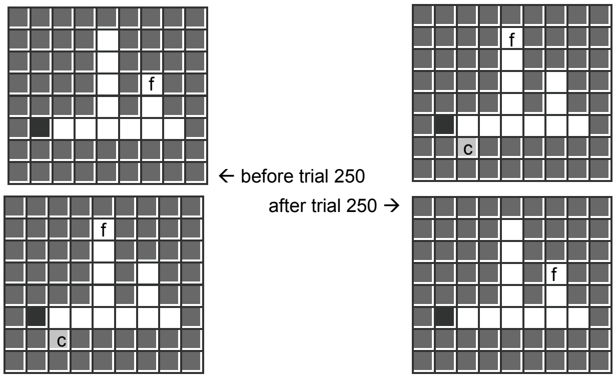

Figure 4.4. Cue-inversion world. The left and right pictures show the possible worlds before and after the cue inversion at trial 250 respectively; ’f’ is food, ‘c’ is cue, black square is the agent.

The second world is based on a typical psychological method in which subjects have to learn to cope with a cue-meaning inversion (see, e.g., Goschke & Dreisbach, 2004). This type of method is used to investigate the effect of an experimental variable, e.g., affect (Goschke & Dreisbach, 2004) on working memory flexibility by measuring reaction time just after the cue-meaning inversion. It is also used to measure adaptation speed to the new cue-meaning relation after having learned the old relation. In the case of our simulated grid world, a cue is coupled to a specific food location, while the absence of that cue is coupled to a different food location. At trial 250, the locations are inversed. This means that whereas before trial 250 the cue indicated to the agent that food is at location 1, after trial 250 the cue ('c' in Figure 4.4) indicates that food is at location 2. We call this world the *cue-inversion* world. In contrast to the switch-invest task, the agent is also reset to its (fixed) starting position when it arrives at the non-goal location (e.g., when the agent has misinterpreted the cue). The non-goal location (empty arm) has a negative reinforcement of r=–0.5.

To test our three hypotheses, we vary the simulation-selection mechanism and analyze how an artificial agent copes with these two worlds. Our agent employs the learning and simulation mechanisms as described in Section 4.3. In total, we define four static simulation-selection mechanisms:

1. No simulation; simulation is off (called *nosim* in the experiments).
2. Simulation of the best predicted action-state pair; t_s=0 (*simbest*).
3. Simulation of the best half of predicted action-state pairs, i.e., t_s=0.5 (*simbest50*).
4. Simulation of all predicted action-state pairs, i.e., t_s=1 (*simall*).

We also define four dynamic simulation mechanisms, introduced in Section 4.3.3. These are:

1. Positive affect = little simulation (select best predicted action-state pairs), and vice versa (*dyn*).
2. Negative affect = little simulation, and vice versa (*dyn inv*).
3. High intensity of affect = little simulation, and vice versa (*dyn intensity*).
4. Low intensity of affect = little simulation, and vice versa (*dyn intensity inv*).

In the switch-to-invest experiments we have used all four static simulation strategies and only the first two dynamic ones. In the cue-inversion experiments we have used all eight simulation strategies. As mentioned earlier, our measure of affect has three parameters that define its behavior. We varied these three parameters, i.e., we varied *f* (sensitivity of affect), *ltar* (the window size of the long term averaged reward that defines "how well is usual"), and *star* (the window size of the short term average reward that defines "how am I doing").

In our switch-to-invest grid-world experiments we varied these according to Table 1, resulting in 30 different affect-parameter settings. In our cue-inversion grid-world experiments we varied these only according to the *f*=1 column in Table 1, resulting in 10 different affect-parameter settings.

Further, in our switch-to-invest experiments we varied the learning rate, α = [0.8, 0.9, 1.0], and the rate at which the model forgets information about the world as defined by the forgetting rate of nodes, θ = [0, 0.01, 0.02, 0.03]. In the cue-inversion experiments α and θ are not varied but fixed at 1 and 0 respectively.

f:	*1*		*1.5*		*2*	
star:	50	100	50	100	50	100
ltar:	200	400	200	400	200	400
	250	500	250	500	250	500
	375	750	375	750	375	750
	500	1000	500	1000	500	1000
	750	1500	750	1500	750	1500

Table 4.1. Possible *ltar*, *star*, and *f* combinations as they are used in the first set of experiments with the agent in the *switch-to-invest* task.

4.5 Experimental Results

We first describe the results obtained with the switch-to-invest grid world, after which we describe the results obtained with the cue-inversion grid world. Data was analyzed as follows. To investigate the effect of learning rate, α, forgetting rate, θ, and simulation strategy we compare between results of different <α ,θ, *simulation strategy*> configurations. Static simulation strategies have been executed 200 times per <α ,θ, *simulation strategy* > configuration, e.g., the simulate-best strategy has been executed 200 times for every <α ,θ> combination. These 200 runs are the basis for further analysis. Dynamic simulation strategies have been executed 15 times per <α ,θ, *f, ltar, star, simulation strategy*> configuration. For every <α ,θ, *simulation strategy*> configuration, the resulting runs for all of its <*f, ltar, star*> settings is aggregated. For example in the switch-to-invest experiments, for α = 1, θ = 0, and strategy=*dyn* we aggregated all 15 x 30 (*nr of runs* times *nr of affect-parameter settings*, respectively) runs into 450 runs. These runs are the basis for further analysis.

In the cue-inversion experiments the same aggregation protocol was used, but, as mentioned above, here we use only one <α ,θ> configuration and we vary only *star* and *ltar* (not *f*). Further, we used 50 runs per <α ,θ> configuration resulting in 50 x 10 runs = 500 runs being aggregated for only one setting (α = 1 and θ = 0).

We aggregated the data as our goal is to investigate the effect of affective control of simulation selection *in general*, not to find specific values that "work" for the agent. We did not seek to optimize any parameter but to investigate different relations between affect and simulation selection. Between simulation strategies we compare:

- A measure for the behavioral *effort* involved in completing a run (i.e., learning the complete task) for each specific simulation strategy. Effort is calculated by *first averaging trial-length in steps over all trials for each run, resulting in an effort for that run.* This is our unit of measurement for statistical analysis (e.g., if there are 450 runs for one strategy, we have 450 measures of effort to use in our statistical analysis for that strategy). To *display* the average effort for a certain simulation strategy, we *average over the measure of effort for all runs for that strategy*. For example in a static selection mechanism ($\alpha = 1$ and $\theta = 0$), the displayed effort equals the mean number of steps needed for one trial over all 500 trials in all 200 runs resulting in, e.g., 20 steps. For a dynamic simulation mechanism the average is constructed in the same way using aggregated runs for every $<\alpha\ ,\theta>$ configuration instead. The Wilcoxon ranked-sum test (non-parametric, we cannot assume normality) is used to compare effort between simulation strategies. Comparison is based on sets of effort measures (Switch-to-invest: n=450; Cue-inversion: n=500). For static strategies 450 samples (Switch-to-invest) or 500 samples (Cue-inversion) are pooled from the 200 runs that are available.
- A measure for the total *simulation effort* involved in completing a run, i.e., the same as above but using a trial-length counted in terms of internally simulated action-state pairs. This represents "mental effort" during a task, and as such is linked to energy consumption used to maintain and focus on information in working memory. Again, the Wilcoxon test is used to compare simulation strategies.

To give an informal idea of the learning behavior of the agent, several learning curves of agents are plotted. Learning curves are plots of the *average number of steps taken per trial* and smoothed using a sliding mean (window size = 10) to improve readability.

4.5.1 Results of Experiment 1: Switch-to-invest Task

Results in this specific grid world show that simulation in general has a stable positive effect on learning. This trend is shown by the learning curves[1] in Figure

[1] Note that we do not use error bars in Figure 5. To validate our claims, we statistically compare between simulation strategies the effort involved in completing a run. This is appropriate; a small overall benefit can be considered important, regardless of the standard deviation over trails.

4.5, and more formally in Figure 4.6 showing that *nosim* uses more effort to complete a run than any other simulation strategy ($p<0.001$). The larger the amount of internally simulated interactions, the better the learning result (*simall* costs less effort than *simbest*, $p<0.05$ for all settings except $\alpha=1$ & $\theta\in\{0, 0.01\}$, Figure 4.6). When affect is used to control this amount, performance is better than the static simulation mechanism that simulates the best strategy (a significant difference between *dynsim* and *simbest*, $p<0.05$ for all settings except $\alpha=1$ & $\theta\in\{0, 0.01\}$, Figure 4.6). Interestingly, the size of the effect interacts with the learning rate and forgetting rate. As θ increases, the benefit of simulation also increases, and as α decreases the benefit of simulation increases (Figure 4.6). In terms of size, we did not find important differences between (1) the dynamic strategy that relates negative affect to more simulation and (2) the dynamic strategy that relates positive affect to more simulation. Even though the strategies are each other's inverse, the difference in effort was at most about 5% (Figure 4.7a, shown only for $\alpha=0.8$ & $\theta=0.03$). However, for all $<\alpha ,\theta>$ settings, the average amount of simulation effort was considerably less for *dyn* than for *dyn inv* ($p<0.001$). Further, both strategies simulated considerably less than *simall* ($p<0.001$), while *dyn* used less simulation effort than *simbest50* ($p<0.001$) (Figure 4.7b, shown only for $\alpha=0.8$). Finally, results for $\alpha=0.9$ are not shown, as these appeared to be an interpolation between the results for $\alpha=0.8$ and $\alpha=1.0$.

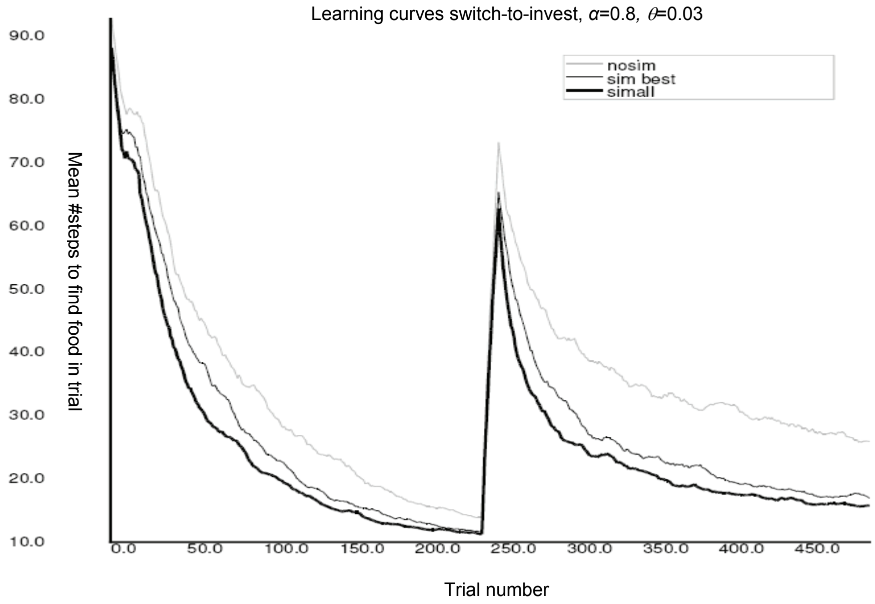

Figure 4.5. Learning curves (smoothed) of non-, best, and all-simulating agents in the switch-to-invest world for α=0.8, θ=0.03. Curves of other strategies are approximately in-between best and all.

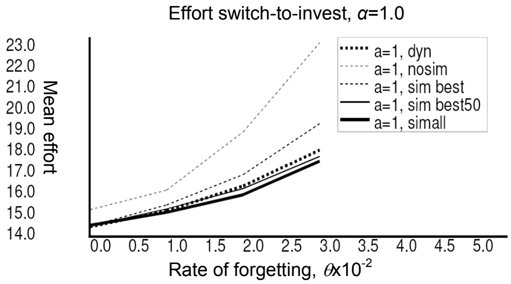

Figure 4.6a. Effort for different simulation strategies in the switch-to-invest task with a learning rate, α, equal to 1.0

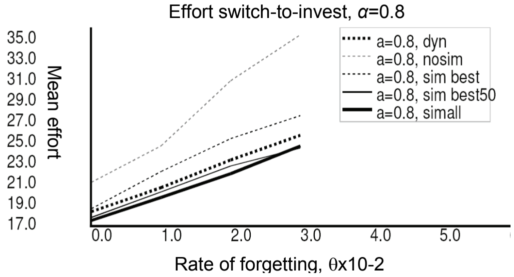

Figure 4.6b. Effort for different simulation strategies in the switch-to-invest task with a learning rate, α, equal to 0.8

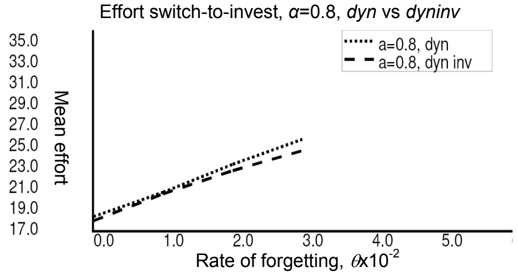

Figure 4.7a. Small difference in effort between dynamic and inverse-dyn simulation strategies.

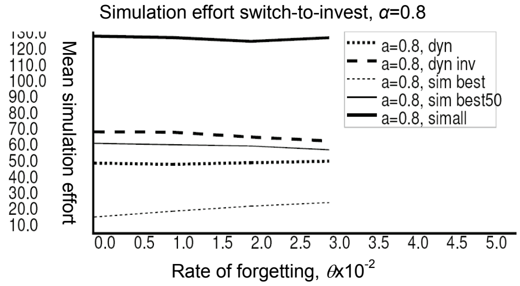

Figure 4.7b. Difference in simulation effort between simulation strategies.

4.5.2 Discussion of the Switch-to-invest Task Results

The fact that more simulation results in better performance is not surprising. Internal simulation as an anticipatory heuristic can use more knowledge if it selects more potential next interactions. Thereby, it influences final action selection in a more balanced way. Interestingly, there is an interaction effect produced by learning rate, forgetting rate and simulation. Regarding the learning rate this effect is easily explained. As internal simulation enables the agent to "look ahead" one step, predicted values can be temporarily propagated back. Even though the model does not learn based on simulation (i.e., nodes, their value, reward and statistic are not permanently updated due to simulation), simulation has an immediate benefit for action selection, as more information is temporarily available. If the learning rate is high (α≈1.0), this effect is minimized: at every step the agent takes, the lazy update rule propagates future values back in full, so simulation cannot add a lot of future value information. However, if the learning rate is small(er) (e.g., α=0.8), the future value is not propagated in full. Now, internal simulation can temporarily propagate values that were not yet propagated in full, and the action-selection mechanism can benefit from the extra information provided by simulation. This phenomenon causes a performance increase due to simulation in lower learning rate settings.

It is not yet clear from our experiments what causes the interaction between forgetting rate and simulation, although it is clear that it can not be simulation per se, as simulation does not change the model's statistics. A possible explanation is that simulation in general forces the agent to use known interaction patterns more often than new or less-tried patterns. As such, simulation actually reduces the probability of forgetting useful interactions. This could help solving the maze with a forgetful long-term memory. This requires further investigation in future research.

The fact that the two dynamic simulation strategies tested (a) do not differ in terms of learning performance, (b) perform at about the same level as the static simulation strategy that simulates all potential next interactions, and (c) use a considerably reduced amount of simulation compared to this static *simall* strategy, indicates two things: (1) dynamic adaptation is beneficial as it reduces simulation needs (an interesting result), and (2) it does *not* matter if positive affect implies more simulation or less, as the two dynamic simulation strategies result in less simulation *and* better learning performance. If the latter is indeed the case, this implies one of the two following possibilities: (I) affect has nothing to with the result. Instead, the average amount of simulation is responsible for the increase in learning performance. This possibility is supported by our results, as the *dyn inverse* strategy uses more simulation than *dyn* (Figure 4.7b) and seems to perform slightly better than the latter (Figure 4.7a). On the other hand, it could also imply that (II) affect *does* have to do with the result, but both relations—i.e., positive-affect = more-simulation and positive-affect = less-simulation—are wrong. This is possible if the relation instead is: higher-intensity-affect=more-simulation. We study this in the second experiment, and use the intensity-of-affect based simulation strategies. In this experiment we use the second grid world, i.e., the cue-inversion world.

4.5.3 Results of Experiment 2: Cue-inversion Task

Results in this grid world show the following. The *simbest* static simulation strategy does not have a large positive effect (even though the effect is significant $p<0.01$), contrary to the results in the first experiment where the effect was more pronounced. However, *simall*, *simbest50* as well as all dynamic simulation strategies do have an important positive effect ($p<0.001$); effort is reduced with 0.6 to 1 step per trial. Thus, a moderate positive influence of simulation on learning performance exists. Note that the smaller effects of simulation in general, as compared to the previous experiment, are due to the fact that in this experiment $\alpha=1$ and $\theta=0$. This confirms our explanation of interaction effects between simulation, α and forgetting rate in the discussion of the previous experiment.

Again, dynamic strategies are quite close to the *simall* strategy in terms of learning performance (Figure 4.8a): the only significant difference in effort is between *simall* and *dyn intensity* (p<0.01). However, dynamic strategies use considerably less simulation effort to get to this increased level of performance (Figure 4.8b, all strategies use less simulation than *dynall,* p<0.001). An important difference in effort exists between the two intensity-based dynamic simulation strategies. The *dyn intensity inverse* strategy (i.e., if affect is neutral, 0.5, simulate a lot, while if affect is extreme, 0 or 1, simulate little) has a better performance than *dyn intensity* (p<0.001, Figure 4.8a), but also uses a lot more simulation (p<0.001).

Last, we plot the average behavior (over 50 runs) of our measure for artificial affect as it is influenced by *ltar* and *star*. A large long term window to calculate the agent's measure of comparison based on reward (i.e., "what I am used to") results in less noisy affect (Figure 4.10). A small short term average (i.e., "how am I doing") results in a faster affective reaction to the cue-inversion (inset).

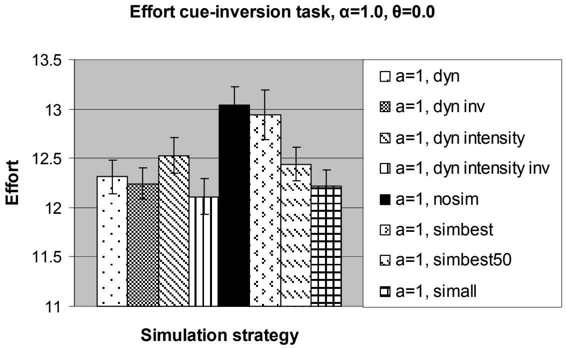

Figure 4.8a. Difference in effort between dynamic and static simulation strategies. Error bars show 95% confidence interval.

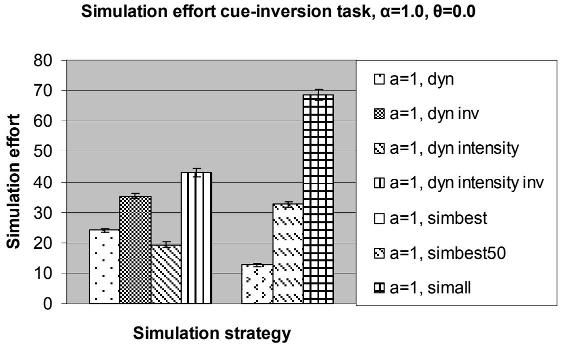

Figure 4.8b. Difference in simulation effort between static and dynamic strategies. Error bars show 95% confidence interval.

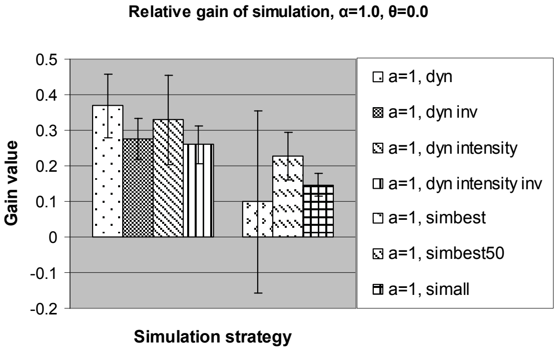

Figure 4.9. Gain of simulation strategies (details in text). Error bars show 95% confidence interval.

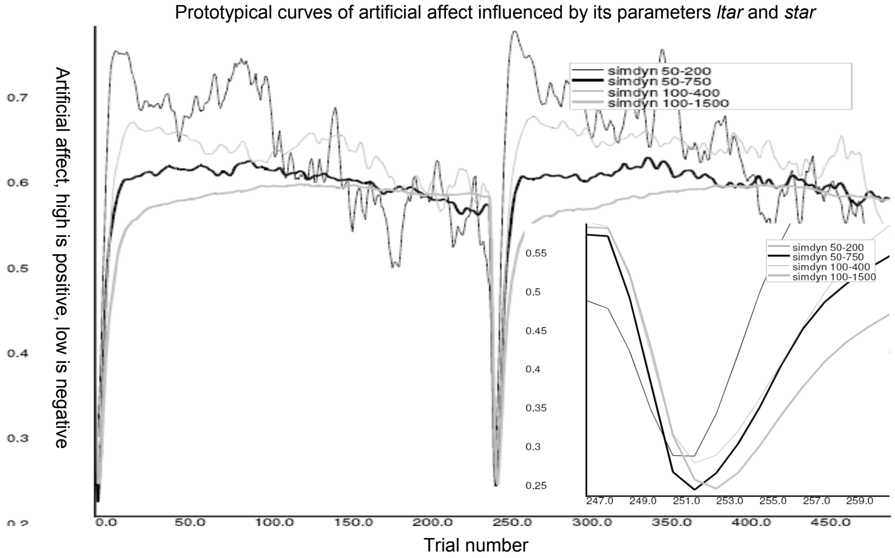

Figure 4.10. Depicted are affect curves for different settings (not smoothed). Inset is a detail of artificial affect at the cue inversion. Note that *star*=50 has the "dip" earlier than *star*=100.

4.5.4 Discussion of the Cue-inversion Task Results

The fourth dynamic control strategy based on the inverse intensity of affect (*dyn intensity inv*) results in a better performance than the third, intensity based, control strategy. Again, this inversed version (i.e., neutral affect results in a lot of simulation and extreme affect in a little) uses more simulation on average. Thus, this result does not rule out the possibility that the average amount of simulation is responsible for the learning performance increase as opposed to affective control. We need to control for the average amount of simulation. To do so, we defined the *gain* ratio, a measure that calculates how much effort reduction a strategy gives relative to no simulation, weighted by the amount of simulation effort:

$$gain_i = (effort_{non} - effort_i)/(sim\_effort_i / effort_i) \qquad (4.11)$$

where $effort_i$ equals the effort for a certain simulation strategy *i*, $effort_{non}$ equals the effort of the *nosim* strategy and sim_effort_i equals the simulation effort for a certain strategy *i*. Such a gain factor is a plausible measure to evaluate and compare simulation strategies: one is interested in the efficiency of simulation, not just the absolute result. As simulation—i.e., information maintenance in working memory—costs resources, the question is which strategy uses these resources best. When we compared the gains for the different simulation strategies, a different picture emerged (Figure 4.9). Simulating all is not very efficient compared to dynamic strategies. Interestingly, our original coupling of affect and amount of simulation seems most promising (as proposed, but not yet confirmed in Broekens and Verbeek, 2005). This is the only strategy of which the gain confidence interval does not overlap with either *simall* or *simbest50*. This means that, although the relation “positive affect equals less simulation and negative affect equals more simulation” is not the best one in terms of effort reduction, it is the optimal one in terms of *relative gain when considering the amount of simulation needed for that effect*.

4.6 General Discussion

We now discuss our approach in a broader context. We first ground our approach more firmly, and relate our work to the work of others. Finally we present some directions for future research.

4.6.1 Model Grounding

Our findings are compatible with psychological findings that show that both positive and negative affect influence learning in a beneficial way (Craig et al., 2004; Dreisbach & Goschke, 2004; Rose et al., 1999). We found that learning benefits the most when positive affect relates to less simulation and negative to more simulation. As such, our findings indicate that positive affect is associated with less diverse thoughts when a task has successfully been learned, while negative affect is associated with diverse thoughts when a task is confusing or changing. Our findings support the studies by Rose et al. (1999) who find that broad attention is associated with faster learning and neutral but not positive affect, when a new task has to be learned. Our findings are also consistent with the relation that has been found between subclinical depression and defocused attention (von Hecker & Meiser, 2005). In agreement with these authors, we would like to stress that our results do not necessarily argue for a "positive affect equals reduction of capacity" view. More selective maintenance of information is not the same as a reduction of capacity. Selectivity of maintenance in Working Memory (WM) that depends on affect can be an adaptive strategy to cope with the changing world around us, without enforcing any capacity constraints.

In our approach, internal simulation influences action-selection in a way that is compatible with the Somatic Marker Hypothesis (SMH) (Damasio, 1994). In short, the SMH states that somatic (i.e., of the body) signals are coupled with representations of situations and thereby function as a value signal that enables the organism to filter potential behaviors. As a result, some of these potential behaviors are selected for conscious contemplation in working memory while others are not. Our threshold determines how discriminating our simulation-selection mechanism is, thereby selectively allowing some anticipated behaviors to enter working memory and influence future behavior. Of course we do not argue that we have an embodied approach; our agent is quite disembodied. However, our action-state value *v* can be interpreted as a simulated marker, as it accumulates future values of potential situations. As such, it is an abstraction of the somatic signal that, in an embodied modeling approach and in nature, is grounded in the body. We argue that our mechanism of simulated interaction selection, and thus selection of WM content, is compatible with the mechanism by which somatic markers are used to prune large amounts of thoughts. Both mechanisms prioritize different anticipated behaviors based on a comparison of their markers. Only potential behaviors (thoughts) that have highly positive markers—or *strong* markers, if the *intensity* of artificial affect is used as simulation-selection threshold (cf. Section 4.3.3)—are able to influence future behavior by temporarily transferring a portion of their own marker value to the marker value of considered actions (see also Damasio, 1994). In our model, transfer of marker values is a natural consequence of simulating a particular future interaction (see Step 1 – 5, Section 4.3.2).

Concerning the relation between our model and the Simulation Hypothesis, several similarities are particularly important. Hesslow (2002) states that fundamentally new mechanisms should not be needed for internal simulation of behavior. The only mechanism we introduce is an interaction feedback loop to the RL model. We do not introduce a conscious reasoning process or a central intelligence that enables planning. Compared to such measures, our addition is just a minor change to the overall agent architecture, and comparable with the addition of a feedback connection in neural network models that investigate internal simulation (van Dartel & Postma, 2004; van Dartel, Postma & van den Herik, 2005). Further, our mechanism for simulation selection is very similar to that of action selection: the RL model is used in the same way in both the simulation (cognitive) and non-simulating (reactive) setting; simulation selection uses the action-selection component; and the representations used for simulation are the same as those used for action.

Hesslow (2002) also states that internal simulation of behavior uses the same sensory-motor mechanisms as actual behavior, and therefore uses similar sensory-motor encoding. Our interactions encode features of the world coupled with actions, and our model uses these same interactions for simulation. More importantly, in our model, simulation influences action indirectly: an influence that results *only* from making use of the same mechanisms needed for action. This is very compatible with the Simulation Hypothesis stating that simulation and action are tightly coupled. Our mechanism for influencing action selection is therefore a useful addition to the Simulation Hypothesis by postulating a potential mechanism by which internal simulation could influence action: i.e., simulation temporarily biases next actions *because* the simulation mechanism and action mechanism overlap and therefore simulation activates potential next actions to some extent, resulting in the “markers” of the simulated consequences to be temporarily attached to these next actions.

4.6.2 Related Work

To show that simulation in our model can indeed be seen as an instantiation of simulation as meant by the Simulation Hypothesis we compare it with the models by van Dartel and Postma (2005), van Dartel et al. (2005) and Ziemke, Jirenhed and Hesslow (2005). These models use a genetic algorithm to train a neural network to produce predictions of future states one time step ahead. These predictions are used to bias perception of the current state (van Dartel), or explicitly used as input to the neural network controller to enable “’blindfolded’ corridor following behavior” based on these simulated next states (Ziemke). Although our action-state encoding and learning mechanism are different, our overall architectural approach is similar, especially to the work of van Dartel and colleagues. Simulation in the latter work is modeled as follows. A copy of the output layer (encoding actions) of the neural network is projected to the input layer. This output copy consequently influences perception, and influences action selection. The feedback from this copy to the input represents a simulated next state as predicted by the model (van Dartel & Postma, 2005). These authors explicitly suggest that in their model internal simulation "serves the function of building up sufficient activation in the neurocontroller to produce a certain move". This is equivalent to what happens when in our model future interactions are simulated, as these simulated interactions bias the "markers" of current potential actions and as such can help certain actions to be executed. The work of Ziemke et al (2005) is a bit different. They train an "input prediction layer" to predict the next observed state based on the current one. This prediction is used as input to an already trained sensory-motor network responsible for collision-free corridor following behavior. The predicted state is used as real input to the sensory-motor network such that the agent as a whole walks through the corridor based on mental simulations of interaction with the corridor, i.e., it is walking "blind-folded". The characteristic difference between this model and our model is that Ziemke et al. use the predicted next state as input for action-selection, while in our model the simulated input is used as a bias, as in the model by van Dartel. However, from an architectural point of view, the three models are all instantiations of the Simulation Hypothesis: the models internally simulate predicted interaction with the environment in order to influence actual interaction, while using the same encoding and the same mechanisms for both real and simulated interaction.

Simulation in our approach is to some extent similar to planning in *Dyna* (Sutton, 1990). However, several important differences exist. First, our model learns multiple MDPs in parallel and uses all of these MDPs in action selection. Second, anticipatory simulation in our model (cf. planning in *Dyna*) is always a one-step forward simulation from the current state, not a simulation of a random state. This reflects our choice of basing the model on anticipatory simulation of behavior, and not on planning or dynamic programming in general. As a result, the potential of simulation in our model is more limited. Third, our model can only simulate actions it has tried already, effectively restricting the exploration potential of broad simulation. This is the most important reason why simulating all potential next action-states is not really equivalent to exploration. Our agent cannot really explore mentally, it can only consider the many known future options, in contrast to *Dyna* in which untried actions can be simulated. However, in order to do so, *Dyna* requires a non-empty world model to start learning (Sutton, 1990). We have chosen to start learning with a completely empty model. Therefore we could not simulate untried actions, at least not without making major changes to the representations of action-state pairs and transitions between them. Finally, simulation in our model has a temporary effect by biasing the predicted values of next states and thereby influencing action selection. In *Dyna*, planning can actually change the evaluation and policy functions.

Notwithstanding these differences, our method of internal anticipatory simulation of states replicates some of the results obtained with *Dyna* (Sutton, 1990), of which the most relevant in the context of the presented results is that simulation (and more simulation rather than less) has a positive effect on learning speed.

Our results show that internal anticipatory simulation of just one step ahead is beneficial to artificial adaptive agents, even if simulation does not alter the long-term knowledge of the agent. The influence of simulation is mediated by the action selection mechanism of the agent. Simulation introduces a temporary bias to the values predicted by the model. This approach is similar to the one proposed by Gadanho (2003). In her RL based adaptive system, however, stochastic action-selection is biased by a fixed value produced by a rule-based cognitive system. In contrast, in our system this value is dependent on the predicted states and the cognitive process is not separated from the adaptive system. We did not separate these systems as the Simulation Hypothesis is underlying our approach. As internal simulation of behavior is based on existing sensory-motor mechanisms, it made sense to investigate the benefit of anticipatory simulation using as many functions as possible already provided by our RL model.

4.7 Conclusion

Using a computational model based on Reinforcement Learning, we have investigated affective control of anticipatory thoughts, where thoughts are defined as internal simulation of potential next behavior (Cotterill, 2001; Hesslow, 2002). We have introduced a simulation-selection mechanism that is controlled by affect and selects anticipatory behaviors for simulation from the predictions of the RL model used by the agent. The selected anticipatory behaviors are used to bias the predicted values of next action-state pairs. Action selection is over these biased pairs, thereby influenced by the simulated anticipations. Based on experiments with adaptive agents that learn two nondeterministic partially observable grid worlds we conclude that (1) anticipation has an adaptive benefit and (2) affect can be used to control the amount of simulation. The results show that affective control reduces the amount of simulation needed to get a performance increase due to simulation.

The positive effect of internal simulation has been shown to exist for two non-deterministic partially observable worlds, and already has been shown to exist in other worlds (Broekens, 2005). However, selecting all possible next action-state pairs for simulation provides quite some computational overhead, or, in more biological terms, consumes a considerable amount of energy to maintain stable representations in working memory (WM) that can be used to construct anticipatory associations. In this study we have shown that affect can regulate the amount of anticipatory simulation in such a way that learning is still improved considerably. Although it is difficult to generalize from computational experiments that contain many variables, in terms of WM-affect relation our results indicate that affective control of the amount of anticipatory thoughts in WM enables an adaptive agent to make more efficient use of WM.

The most beneficial relation between affect and internal simulation is observed when positive affect decreases the amount of simulation towards simulating the best potential next action, while negative affect increases the amount of simulation towards simulating all potential next actions. Ergo, agents "feeling positive" can think ahead in a narrow sense and free-up working memory resources, while agents "feeling negative" must think ahead in a broad sense and maximize usage of working memory. Our results are consistent with several psychological findings on the relation between affect and learning, and contribute to answering the question of *when* positive versus negative affect is useful during adaptation. Furthermore, our results show that simulation selection is a useful extension to action selection, specifically in the context of the Simulation Hypothesis (Hesslow, 2002).